Weekly report

1 Done

1.1 Reading

1.1.1 Maximizing Bichromatic Reverse Spatial and Textual k Nearest Neighbor Queries（Farhana M. Choudhury）

This paper resolves the problem of maximizing bichromatic reverse k nearest neighbor queries (BRkNN) by new spatial-textual indexes, namely Min-max IR tree (MIR-tree). The spatial part of MIR-tree is same as IR-tree (as Figure 1), besides, each term in MIR-tree is associated with both the maximum similarity and minimum similarity with the documents (Figure 2).

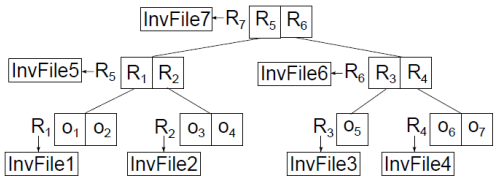


Figure 1

Term	InvFile 1	InvFile 2	InvFile 3	InvFile 4	InvFile 5	InvFile 6	InvFile 7
t_1	$(o_1,1,1)$	$(o_3,5,5)$	$(o_5,4,4)$	$(o_6,1,1),(o_7,2,2)$	$(R_1,1,0),(R_2,5,0)$	$(R_3,4,4),(R_4,2,1)$	$(R_5,5,0),(R_6,4,1)$
t_2	$(o_1,4,4)$	-	$(o_5,1,1)$	-	$(R_1,4,0)$	$(R_3,1,1)$	$(R_5,4,0),(R_6,1,0)$
t_3	-	$(o_3,5,5)$	-	$(o_6,1,1)$	$(R_2,5,0)$	$(R_4,1,0)$	$(R_5,5,0),(R_6,1,0)$
t_4	$(o_2,1,1)$	$(o_4,2,2)$	-	$(o_7,3,3)$	$(R_1,1,0),(R_2,2,0)$	$(R_4,3,0)$	$(R_5,2,0),(R_6,3,0)$

Figure 2

Related requests including location and terms can be transformed into Modified IUR-tree as well.

The main algorithm (Joint Top K) is shown below.

Algorithm 1: JOINT_ TOPK(MIR-tree, U, k)

```
1.1  Output: The top-k objects of all users
1.2  u_s.l ← ∀u∈U MBR(u.l); u_s.dUni ← ∀u∈U ∪(u.d); u_s.dInt ← ∀u∈U ∩(u.d)
1.3  Initialize max-priority queue PQ, RO, min-priority queue LO
1.4  E ← MIR-tree(root)
1.5  ENQUEUE(PQ, E, LB(E, u_s))
1.6  while PQ ≠ ∅ do
1.7      E ← DEQUEUE(PQ)
1.8      if E is leaf then
1.9          if |LO| < k then
1.10             ENQUEUE(LO, E, LB(E, u_s))
1.11             if |LO| = k then
1.12                 RS_k(u_s) ← LB(TOP(LO), u_s)
1.13         else if UB(E, u_s) ≥ RS_k(u_s) then
1.14             ENQUEUE(LO, E, LB(E, u_s))
1.15             Obj ← DEQUEUE(LO)
1.16             RS_k(u_s) ← LB(Obj, u_s)
1.17             if UB(Obj, u_s) ≥ RS_k(u_s) then
1.18                 ENQUEUE(RO, Obj, UB(Obj, u_s))
1.19     else
1.20         if |LO| < k or UB(E, u_s) ≥ RS_k(u_s) then
1.21             for each element e of E do
1.22                 ENQUEUE(PQ, e, LB(e, u_s))
1.23 return INDIVIDUAL_TOPK(U, u_s, LO, RO)
```

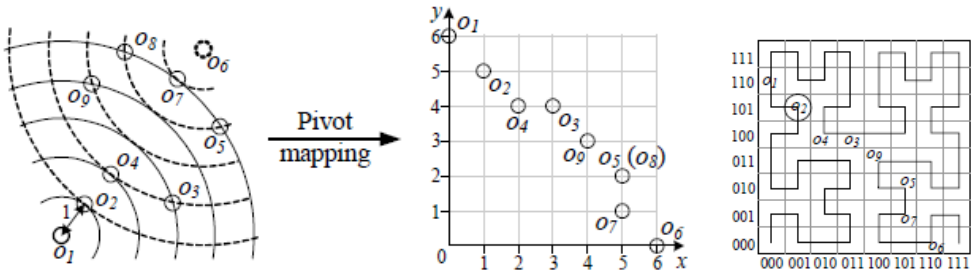
K number of objects with the best lower bounds found so far are stored in a min-

priority queue LO. Those key is upper bound similarity score is stored in max-priority queue RO. Besides, max-priority queue PQ is created to keep track of the nodes that are yet to be visited.

This paper provides algorithms for candidate location selection and candidate keyword selection. About the candidate keyword selection, it provides approximate algorithm and exact algorithm for the question is NP-hard.

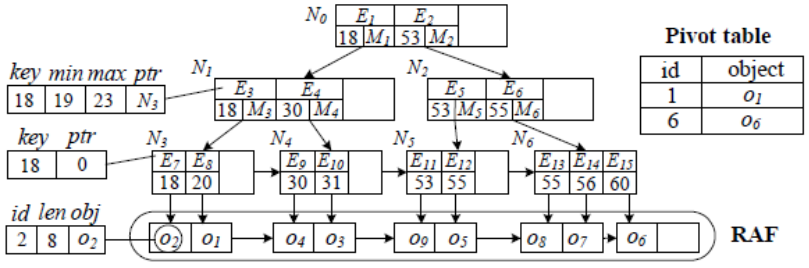
1.1.2 Efficient Metric Indexing for Similarity Search (Lu Chen)

This paper develops a new metric index, namely, Space-filling curve and Pivot-based B$^+$-tree (SPB-tree). SPB-tree picks few but effective pivots to reduce the number of distance computations; uses SFC to cluster data objects into compact regions; utilizes a B$^+$-tree with MBB information as underlying index.

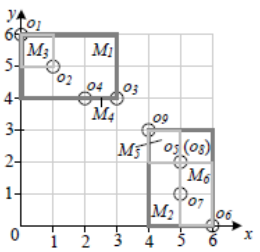


They select pivots from the entire set by HF based Incremental pivot selection algorithm (HFI). Good pivots are usually outliers, so they filter pivots from outliers. Then, they applied Hilbert Curve to mapping the metric space into discrete integers. Note that if the number of cells are too many, the vector space may become too sparse.

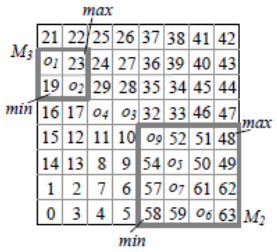
Finally, a tree is built as bellow.



(a) The SPB-tree structure



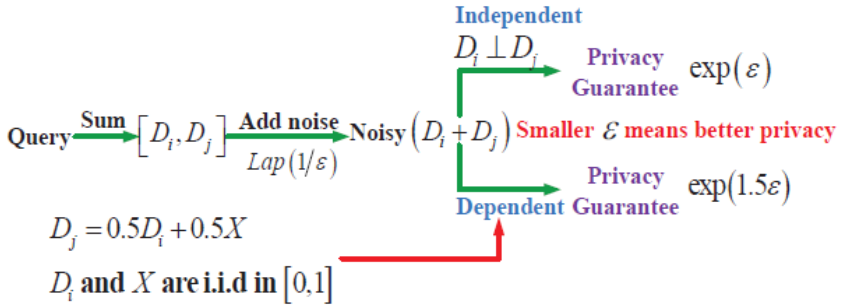
(b) MBBs of an SPB-tree



(c) The MBB representation

1.1.3 Dependence Makes You Vulnerable: Differential Privacy Under Dependent Tuples (Changchang Liu)

As mentioned in my last weekly report, one of the limitation of differential privacy is that it need independence assumption. For example, as shown in the figure below, the privacy guarantee is degraded by dependence between tuples.



Thus, dependent differential privacy is provided for this issue.

- Dependence size L: if any tuple in D is dependent on at most L-1 other tuples.
- Probabilistic dependence relationship R: due to the data generating process.
- Dependent Differential Privacy:

$$\max_{D(L,\mathcal{R}),D'(L,\mathcal{R})} \frac{P(\mathcal{A}(D(L,\mathcal{R}))=S)}{P(\mathcal{A}(D'(L,\mathcal{R}))=S)} \leq \exp(\epsilon)$$

- Dependence coefficient ρ_{ij}: evaluates the dependence relationship between D_i and D_j from the privacy perspective. Note that it is asymmetric.

$$\rho_{ij} = \frac{Dependent\ \ Indistinguishability}{Self\ \ Indistinguishability} = \frac{\max_{\mathbf{d}_i} \log \left\{ \sum_{\mathbf{d}_j} P(D_j = \mathbf{d}_j | D_i = \mathbf{d}_i) \exp\left(\frac{\|\mathbf{d}_j - \mathbf{d}_j^*\|_1}{\sigma(\epsilon)} \right) \right\} \sigma(\epsilon)}{\Delta D_j}$$

Based on the dependence coefficient, the dependent sensitivity to multiple users is:

$$DS_i^Q = \sum_{j=C_{i1}}^{C_{iL}} \rho_{ij} \Delta Q_j$$

The dependent sensitivity for publishing any query Q over a dependent dataset is:

$$DS^Q = \max_i DS_i^Q$$

For any query function Q over an arbitrary domain D with dependent tuples, the mechanism A gives such dependent differential privacy.

$$\mathcal{A}(D) = Q(D) + Lap(DS^Q/\epsilon)$$

1.2 New Idea

I discussed with Jia-Kai about if we can combine prior idea with differential privacy. He thought maybe we can locate privacy problem with k-anonymity, l-diversity, t-closeness and protect privacy by differential privacy. However, we are not

sure if the differential privacy approach can add local noise rather than global noise. Besides, I think if we choose differential privacy, cluster approach is not necessary for better privacy preservation and pattern maintaining.

Besides, how to compare patterns is another problem. Which kinds of patterns may appear is not sure.

2 To Do

2.1 Final review